# Termination of Order-sorted Rewriting *

Isabelle GNAEDIG
*INRIA Lorraine - CRIN CNRS*
*Technopôle de Nancy-Brabois - BP 101*
*54600 Villers-lès-Nancy FRANCE*
*Phone number: 83 59 30 13*
*E-mail: gnaedig@loria.fr*

No Institute Given

**Abstract.** In this paper, the problem of termination of rewriting in order-sorted algebras is addressed for the first time. Our goal is to perform termination proofs of programs for executable specification languages like $OBJ3$. An extension of Lexicographic Path Ordering is proposed, that gives a termination proof for order-sorted rewrite systems, that would not terminate in the unsorted case. We mention also, that this extension provides a termination tool for unsorted terminating systems, that usual orderings cannot handle.

## 1 Introduction

The significance of order-sorted algebras is constantly increasing. First, specification languages with an algebraic semantics based on many-sorted algebras were developed, e.g. $OBJ$ 1 [8], $PLUSS$ [1] or $LARCH$ [9]. The sorted model is indeed more suitable than the homogeneous one to treat computable objects in software problems, since programs manipulate several kinds of objects.

Switching from many-sorted models to order-sorted models allows inclusion of sorts, and inheritance of properties. In this approach, the set of sorts is partially ordered. The ordering on sorts corresponds to domain inclusion in models. Operations and properties from a superset can be applied to a subset, for example + on reals to integers. Executable specification languages like $OBJ3$ [13] or $TEL$ [14] have an algebraic semantics based on order-sorted algebras.

In $OBJ3$, order-sorted specifications have an operational semantics based on rewriting. Rewriting tools, well-known and well developed in homogeneous algebras, extend to order-sorted algebras up to modifications concerning equivalence between equational deduction and rewriting, and also confluence, decidability of local confluence and completion. However, the problem of termination, also called strong normalisation [2], was never tackled in this specific framework although it is a key for the operational semantics. Indeed, together with local confluence, it insures the existence of normal forms. In addition, it is especially important in a completion procedure [6], and last but not least, tools for proving termination of rewriting are, in this context, nothing but means for ensuring termination of programs.

A first approach to termination of order-sorted rewriting consists in forgetting sorts of variables and ranks of operators. Overloaded operators are then considered as one operator. If a rewrite system terminates on unsorted terms, so it is on order-sorted ones. It is the idea retained in $ELIOS\_OBJ$, a version of $OBJ3$ enriched by an order-sorted completion procedure [5]. A Lexicographic Path Ordering (LPO in short) [12] on unsorted terms is used in the implementation.

But such a method is not enough since there are systems that do not terminate if sorts are forgotten; they instead terminate because of the sorts. Let us illustrate that by a natural example given using an $OBJ$-like syntax.

* The original publication is available at www.springerlink.com https://link.springer.com/chapter/10.1007/BFb0013818

This example defines a predicate $is\_even$ over the integers. The idea is to define it over the positive numbers first and to extend it to negative numbers by using the operation "$opposite$".

$obj$ **EVEN** $is$
$sorts\ Zero\ NzNeg\ Neg\ NzPos\ Pos\ Int\ Bool.$
$subsorts\ Zero < Neg < Int.$
$subsorts\ Zero < Pos < Int.$
$subsorts\ NzNeg < Neg.$
$subsorts\ NzPos < Pos.$

$op\ 0 :\rightarrow Zero.$
$op\ s : Pos \rightarrow NzPos.$
$op\ p : Neg \rightarrow NzNeg.$
$op\ true :\rightarrow Bool.$
$op\ false :\rightarrow Bool.$
$op\ is\_even : Int \rightarrow Bool.$
$op\ is\_even : NzPos \rightarrow Bool.$
$op\ is\_even : NzNeg \rightarrow Bool.$
$op\ opposite : NzNeg \rightarrow NzPos.$

$rr\ is\_even(0) \rightarrow true.$ (1)
$rr\ is\_even(s(0)) \rightarrow false.$ (2)
$rr\ (\forall\ x : Pos)\ is\_even(s(s(x))) \rightarrow is\_even(x).$ (3)
$rr\ (\forall\ y : NzNeg)\ is\_even(y) \rightarrow is\_even(opposite(y)).$ (4)
$rr\ opposite(p(0)) \rightarrow s(0).$ (5)
$rr\ (\forall\ y : NzNeg)\ opposite(p(y)) \rightarrow s(opposite(y)).$ (6)
$jbo$

At run time, the set of rewrite rules is used to reduce expressions to their normal form. If the set of rules is confluent and terminating, the normal forms always exist and are unique. In the case of an expression of the form $is\_even(s)$, the normal form is either $true$ or $false$. Let us have a look at the termination proof of this set of rules.

The rules $(1), (2), (3), (5)$ *and* $(6)$ can be oriented with a LPO using the following precedence: "$is\_even$" $>$ "$true$", "$is\_even$" $>$ "$false$", "$opposite$" $>$ "$s$". Equation (4) however, does not terminate since it generates the infinite rewrite chain

$$is\_even(y) \rightarrow is\_even(opposite(y)) \rightarrow is\_even(opposite(opposite(y)))...$$

However, using order-sorted arguments, we can see that the rule

$$is\_even(y) \rightarrow is\_even(opposite(y))$$

terminates. Indeed, in the left-hand-side, the argument of "$is\_even$" is $y$ of type $NzNeg$, whereas $opposite(y)$ is of type $NzPos$ in the right-hand-side. So the left-hand-side can never match the right-hand-side, as any subterm derived from this term, hence the rewrite system terminates.

This example shows that classical termination tools do not take into account the whole order-sorted termination problem. Our aim is then to propose orderings that handle examples of the previous kind, where the sorts play a main role. For that, we work on a disambiguated presentation of the specification, where operators in terms are labeled by a rank.

Let us consider Equation (4) again, and label the operator "$is\_even$" with a rank. With their new label, the occurences of the operator "$is\_even$" become different in the left-hand-side and in the right-hand-side. With respect to the sort of its argument, the operator "$is\_even$" is disambiguated; its rank is "$is\_even$" : $NzNeg \to Bool$ in the left-hand-side. Similarly, the top operator of the right-hand-side has the rank "$is\_even$" : $NzPos \to Bool$. If now an LPO is used on the order-sorted algebra, and if distinct operators have different ranks, the precedence "$is\_even$" : $NzNeg \to Bool$ $>$ "$is\_even$" : $NzPos \to Bool$ and "$is\_even$" : $NzNeg \to Bool$ $>$ "$opposite$" leads to the conclusion $is\_even_{NzNeg \to Bool}(y) >_{LPO} is\_even_{NzPos \to Bool}(opposite(y))$. $LPO$ has been singled out from other existing orderings because it is both simple and covers a large number of standard examples.

Note that the influence of the type structure on termination of rewrite systems is known and a famous case is $\lambda$-calculus [10]. But typed $\lambda$-calculus has a hard and specific proof of termination, and we propose here a method available for typed rewrite systems in general.

On the previous example, considering terms and operators without sort arguments corresponds to the ambiguous presentation of the specification $EVEN$. As shown on this example, this presentation is not adequate for termination of rules, since Rule (4) seems to be not terminating at this level. When we give an unambiguous rank to operators (for example "$is\_even$" : $NzNeg \to Bool$ and "$is\_even$" : $NzPos \to Bool$ ), we consider the disambiguated presentation of the specification. There is enough information at this level to ensure termination of Rule (4).

Our aim is to establish rigorously this new $LPO$ on disambiguated terms. This requires to define precisely the concept of disambiguated presentation of order-sorted specifications. We first have to define a rewriting relation on disambiguated terms, on which $LPO$ will be applied in order to prove termination. For that, we introduce the notion of minimal signature, which ensures that the rank of any operator in a term remains the same, when the arguments of the operator are changed by substitution, even if they get a smaller lowest sort. We then have to prove the algebraic properties of the ordering: replacement property, subterm property, stability by substitution. We finally give a termination theorem similar to Dershowitz' one [3], based on Kruskal's theorem on the disambiguated set of terms.

In Section 2, the definitions related to order-sorted algebras and rewriting are recalled, as well as results on disambiguated order-sorted algebras. The existence of a set of lowest disambiguations of terms is pointed out, and a rewriting relation is defined. The definition of the disambiguated presentation of an order-sorted specification is then given and it is shown that termination of a disambiguated term rewriting system implies termination of the corresponding order-sorted one. Section 3 establishes that our extension of $LPO$ is a simplification ordering on the set of lowest disambiguations of terms, provided the signature is minimal. Section 4 establishes the termination theorem in this context. Section 5 gives examples. One of them illustrates that this extension also provides a termination tool for unsorted terminating systems, that usual orderings cannot handle.

## 2 Order-sorted rewriting

In this section, we give basic notions on order-sorted algebras, following [7]. We recall the concept of regularity for a signature, which ensures the existence of a free term algebra, and the existence of a least sort for any term of the algebra.

Then, we introduce order-sorted rewriting, and the notion of sort-decreasing rewrite rules, which is required when rewriting is used as an operational method in languages like $OBJ3$. Indeed, only sort-decreasing order-sorted systems are complete with respect to deduction as defined in [7, 13, 6]. We also introduce the notion of lowest disambiguation of terms, and define a disambiguated rewriting relation on them.

### 2.1 Signatures and carriers

Given an index set $S$, an $S$-sorted set $A$ is a family of sets, one for each $s \in S$; we write $\{A_s | s \in S\}$. Similarly, given two $S$-sorted sets $A$ and $B$, an $S$-sorted function $f : A \to B$ is an $S$-indexed family $f = \{f_s : A_s \to B_s | s \in S\}$. Let us assume a fixed partially ordered set $(S, \leq)$, called the **sort set**.

An **order-sorted signature** is a triple $(S, \leq, \Sigma)$, where $S$ is a sort set, $\Sigma$ is an $S^* \times S$ -indexed family $\{\Sigma_{w,s} | w \in S^*, s \in S\}$ and $(S, \leq)$ is a partially ordered set. Elements of (sets in) $\Sigma$ are called operators. When the sort set $S$ is clear, we write $\Sigma$ for $(S, \Sigma)$. Similarly, when the partially ordered set $(S, \leq)$ is clear, we write $\Sigma$ for $(S, \leq, \Sigma)$.

For operators, we write $f : w \to s$ or $f : w, s \in \Sigma$ for $f \in \Sigma_{w,s}$ to emphasize that $f$ denotes a function with **arity** $w$ and **co-arity** (or value sort) $s$. The pair $w \to s$ is called rank of $f$. An important special case is when $w$ is $\lambda$, the empty string; then $f \in \Sigma_{\lambda,s}$ denotes a constant of sort $s$.

Note that the ordering $\leq$ on $S$ extends to strings of the same length in $S^*$ by $s_1...s_n \leq s'_1...s'_n$ iff $s_i \leq s'_i$ for $i = 1, ..., n$; similarly, $\leq$ extends to pairs $(w, s) \in S^* \times S$ by $(w, s) \leq (w', s')$ iff $w \leq w'$ and $s \leq s'$ .

Let $(S, \leq, \Sigma)$ be an order-sorted signature. A $(S, \leq, \Sigma)$-**algebra** A consists of a family $\{A_s | s \in S\}$ of subsets of A, called the **carriers** of A, and a function $f_A : A_w \to A_s$ for each $f \in \Sigma_{w,s}$ where $A_w = A_{s_1} \times ... \times A_{s_n}$ when $w = s_1...s_n$ and $A_w$ is a one point set when $w = \lambda$, such that:

1. $s \leq s'$ in $S$ implies $A_s \subseteq A_{s'}$ and
2. $f \in \Sigma_{w,s} \cap \Sigma_{w',s'}$ with $s' \leq s$ and $w' \leq w$ implies $f_A : A_w \to A_s$ equals $f_A : A_{w'} \to A_{s'}$ on $A_{w'}$.

We may write $f_A^{w,s}$ instead of $f_A : A_w \to A_s$; moreover, the second condition means that we can often omit the superscript $w, s$ without ambiguity.

### 2.2 Order-sorted term algebra and rewriting

Following [7], we now give the definition of the **order-sorted $\Sigma$-term algebra** $\mathcal{T}_\Sigma$ as the least family $\{\mathcal{T}_{\Sigma,s} | s \in S\}$ of sets satisfying the following conditions:

- $\Sigma_{\lambda,s} \subseteq \mathcal{T}_{\Sigma,s}$ for $s \in S$;
- $\mathcal{T}_{\Sigma,s'} \subseteq \mathcal{T}_{\Sigma,s}$ if $s' \leq s$;
- if $f \in \Sigma_{w,s}$ with $w = s_1...s_n \neq \lambda$ and if $t_i \in \mathcal{T}_{\Sigma,s_i}$ then (the string) $f(t_1...t_n)$ is in $\mathcal{T}_{\Sigma,s}$.

A term is an element of $\mathcal{T}_\Sigma$. Following [11], we denote by $\mathcal{D}(t)$ the set of occurrences of $t$ i.e. the domain of the term $t$ viewed as a partial operation from $\mathcal{N}^*$ to $\Sigma$. We denote by $t_{|\omega}$ the subterm of $t$ at position $\omega$ and by $t[\omega \leftarrow t']$ the result of the replacement by $t'$ of $t_{|\omega}$ at $\omega \in \mathcal{D}(t)$.

We restrict to the class of regular signatures. Essentially, regularity asserts that overloaded operations are consistent under restriction to subsorts, so that each well-formed expression on the function symbols has a least sort.

When there is a finite set of sorts, regularity can be expressed in the following way. An order-sorted signature $\Sigma$ is **regular** iff for any $w_0 \in S^*$ such that there is a $f \in \Sigma_{w,s}$ with $w_0 \leq w$, there is a least $(w', s') \in S^* \times S$ such that $f \in \Sigma_{w',s'}$ and $w_0 \leq w'$.

If the signature is regular, for any $t \in \mathcal{T}_\Sigma$, there is a least $s \in S$, called the **lowest sort** of $t$ and denoted $LS(t)$, such that $t \in \mathcal{T}_{\Sigma,s}$. From now on, we assume that all order-sorted signatures are regular.

An $S$-**sorted variable set** is an $S$-indexed family $V = \{V_s | s \in S\}$ of disjoint sets. A variable $x$ of sort $s$ is also denoted $x : s$. In this paper, variables are always sorted.

Given an order-sorted signature $(S, \leq, \Sigma)$ and a variable set $V$ that is disjoint from $\Sigma$, $(S, \leq, \Sigma(V))$ is defined by $\Sigma(V)_{\lambda,s} = \Sigma_{\lambda,s} \cup V_s$ and $\Sigma(V)_{w,s} = \Sigma_{w,s}$ for $w \neq \lambda$.

Note that if $\Sigma$ is regular, so is $\Sigma(V)$. We can now form $\mathcal{T}_{\Sigma(V)}$ and then view it as a $\Sigma$-algebra; let us denote this $\Sigma$-algebra by $\mathcal{T}_\Sigma(V)$. $\mathcal{V}(t)$ denotes the set of variables of the term $t$. $\mathcal{T}_\Sigma(V)$ is the **free** $\Sigma$-algebra generated by $V$.

An order-sorted **rewrite rule** is a triple $(X, l, r)$ where $X$ is a variable set and $l, r \in \mathcal{T}_\Sigma(X)$ with $LS(l)$ and $LS(r)$ in the same connected component of $(S, \leq)$ and such that $\mathcal{V}(r) \subseteq \mathcal{V}(l)$. We will use the notation $((\forall X) l \to r)$. When the variable set $X$ can be deduced from the context

(typically just the variables occurring in $l$ and $r$, whose sorts can be uniquely determined or have been previously declared), we allow it to be omitted, i.e., we allow unquantified rewrite rules.

For $(S, \leq, \Sigma)$ an order-sorted signature and $X, Y$ two $S$-sorted variable sets, a **substitution** is an S-sorted function $\sigma : X \to \mathcal{T}_\Sigma(Y)$, that can be extended in a unique manner into $\sigma : \mathcal{T}_\Sigma(X) \to \mathcal{T}_\Sigma(Y)$.

Let $R$ be a set of rewrite rules. Let us now define the **order-sorted rewriting relation**. A **match** from a term $t \in \mathcal{T}_\Sigma(X)$ to a term $t' \in \mathcal{T}_\Sigma(Y)$ is a substitution $\sigma$ such that $\sigma t = t'$.

A term $t \in \mathcal{T}_\Sigma(Y)$ **rewrites** to $t'$ with a rewrite rule $((\forall X) l \to r)$ in $R$ at occurrence $\omega$, which is denoted $t \to_Y^R t' = t[\omega \leftarrow \sigma r]$ whenever

1. there is a match $\sigma : X \to \mathcal{T}_\Sigma(Y)$ from $l$ to $t$ at occurrence $\omega$ ($\sigma l = t_{|\omega}$)
2. there is a sort $s$ such that, for $x$ a variable of sort $s$, $t[\omega \leftarrow x]$ is a well-formed term and $\sigma l, \sigma r \in \mathcal{T}_{\Sigma,s}(Y)$.

The symmetric reflexive transitive closure of $\to_Y^R$ is called *order-sorted* replacement of equals by equals. For the notion of order-sorted replacement of equals by equals to be correct and complete with respect to order-sorted deduction, the term rewriting system has to be sort-decreasing [13]. An order-sorted term rewriting system $R$ on $\mathcal{T}_\Sigma$(Y) is **sort-decreasing** iff $\forall t, t' \in \mathcal{T}_\Sigma(Y),\ t \to_Y^R t'$ implies $LS(t) \geq LS(t')$. If the set of sorts is finite or if each sort has a finite number of sorts below it, a sufficient condition for sort-decreasingness of a term rewriting system is given by the following theorem [13].

**Theorem 1.** *An order-sorted term rewriting system $R$ on $\mathcal{T}_\Sigma(Y)$ is* **sort-decreasing** *if each rule of $R$ is, i.e., if for any rule $((\forall X) l \to r)$ in $R$, for any substitution $\sigma$, we have $LS(\sigma l) \geq LS(\sigma r)$.*

From now on, we assume that rewrite systems are sort-decreasing. Remark that, in this case, the condition (2) of the definition of order-sorted rewriting is always fulfilled.

An **order-sorted specification** is a pair $(\Sigma, R)$, where $\Sigma$ is an order-sorted signature and $R$ is an order-sorted term rewriting system defined on $\mathcal{T}_\Sigma(Y)$.

### 2.3 Disambiguations

As suggested in the introduction, we consider the termination problem at the disambiguated presentation level, whose semantics will be described here. We need to consider disambiguated terms, which means that any operator in a term has to receive a rank. The idea of the ordering on terms is based on a precedence on overloaded operators. As shown in the previous example, we want to state "$is\_even$" : $NzNeg \to Bool$ > "$is\_even$" : $NzPos \to Bool$.

For the term $is\_even(y)$, the operator "$is\_even$" can be disambiguated in two forms: since the sort of $y$ is $NzNeg$, "$is\_even$" can be labeled by $NzNeg \to Bool$ or by $Int \to Bool$.

The general idea is to exhibit for each $S$-order-sorted signature $\Sigma$ a corresponding $S$-order-sorted signature $\Sigma^\sharp$ and a set $M$ of axioms such that an order-sorted $\Sigma$-algebra is (up to an isomorphism) "essentially the same" as an order-sorted $\Sigma^\sharp$-algebra satisfying $M$. The disambiguation process produces another order-sorted algebra and terms are reduced using order-sorted matchings and rules whose left-hand sides are order-sorted terms, while their right-hand sides are kept in disambiguated form.

Given an order-sorted signature $\Sigma$ with a sort ordering $(S, \leq)$, the corresponding $\Sigma^\sharp$ has the ordering $(S, \leq)$ as well and a function symbol $f_{w,s} \in \Sigma^\sharp_{w,s}$ for each $f \in \Sigma_{w,s}$ (including the case of constants where $w = \lambda$), given by the morphism $M$ defined using the following rewrite rules:

$$\begin{gathered}\forall (f : s_1 ... s_n \to s), (f : s'_1 ... s'_n \to s') \in \Sigma \\ s_i \leq s'_i \text{ for } (0 \leq i \leq n) \text{ and } s \leq s' \\ \Rightarrow \\ (\forall x_1 : s_1 \in X, ..., \forall x_n : s_n \in X), f_{s'_1 ... s'_n \to s'}(x_1, ..., x_n) \mapsto_M f_{s_1 ... s_n \to s}(x_1, ..., x_n)\end{gathered}$$

The term $f_{s_1 ... s_n \to s}(x_1, ..., x_n)$ is called a disambiguation of the term $f(x_1, ..., x_n)$.

**Lemma 1.** *(Kirchner & all [13]) If $\Sigma$ is regular and $<$ is noetherian on $S$ then $M$ is terminating, Church-Rosser and sort-decreasing.*

Recall that, in this work, any signature is supposed to be regular. Let us introduce now a particular disambiguation for any term of $\mathcal{T}_\Sigma(Y)$, that is the lowest disambiguation. The **lowest disambiguation** $LD(t)$ of a term $t \in \mathcal{T}_\Sigma(Y)$ is the $M$-normal form of any disambiguation of $t$.

Remark that by Lemma 1, $LD(t)$ always exists and is unique for any term $t \in \mathcal{T}_\Sigma(Y)$. The lowest disambiguation can then be seen as a bijection of $\mathcal{T}_\Sigma(Y)$ on the set of all lowest disambiguations $LD(\mathcal{T}_\Sigma(Y))$, denoted from now on by $LDT(Y)$. Note that the signature of $LDT(Y)$ is $\Sigma^\sharp$.

Let us now give the algorithm that computes in a bottom-up process the lowest disambiguation of a term.

- If $t$ is a variable $x \in X_s$, $LD(t) = x$.
- Else $t = f(t_1, ..., t_n)$. $LD(t) = f_{s''_1 ... s''_n \to s''}(LD(t_1), ..., LD(t_n)) = t_0$, where $f_{s''_1 ... s''_n \to s''}$ is the smallest operator such that $LS(t_i) \leq s''_i$.

The previous smallest operator always exists since the signature $\Sigma$ is supposed to be regular. Note that the lowest sort of $t$ $LS(t)$ is $s''$. Moreover, any term $t = f_{s_1 ... s_n \to s}(t_1 ... t_n)$ of $LDT(Y)$ has a unique sort $S(t) = s$.

**Definition 1.** *Let SPEC be an order-sorted specification, whose signature $\Sigma$ is regular, and whose term rewriting system is $R = (l_i \to r_i,\ i \in [1..n])$ on $\mathcal{T}_\Sigma(Y)$. Its* **disambiguated presentation** *is the order-sorted specification whose signature is $\Sigma^\sharp$, whose term rewriting system is $LD(R) = (LD(lhs_i) \to LD(rhs_i),\ i \in [1..n])$ on $LDT(Y)$.*

For proving termination of the term rewriting system of an order-sorted specification, we will prove termination of the corresponding term rewriting system in its disambiguated presentation. We will state for that that the order-sorted rewriting relation is included in the corresponding disambiguated rewriting relation, we give now the definition of.

Let us first define the substitution on $LDT(Y)$, which requires some care as shown on the following example. Let $\Sigma_0$ be the signature:

$$
\begin{aligned}
&sorts\ Nat\ Int.\\
&subsorts\ Nat < Int.\\
&op + : Nat\ Nat \to Nat.\\
&op + : Int\ Int \to Int.\\
&var\ x : Int.\\
&var\ y : Nat.
\end{aligned}
$$

The terms $t = x +_{Int\ Int \to Int} y$ and $t' = y +_{Nat\ Nat \to Nat} y$ are in $LD(\mathcal{T}_{\Sigma_0}(Y))$. Let $\sigma$ be the substitution $\sigma : \ x \mapsto y +_{Nat\ Nat \to Nat} y$. We have $\sigma t = (y +_{Nat\ Nat \to Nat} y) +_{Int\ Int \to Int} y$, which is not in $LD(\mathcal{T}_{\Sigma_0}(Y))$. For the instantiated term to be in $LD(\mathcal{T}_{\Sigma_0}(Y))$, the top symbol $+_{Int\ Int \to Int}$ has to be replaced by $+_{Nat\ Nat \to Nat}$. So the substitution is not an internal operation on $LD(\mathcal{T}_{\Sigma_0}(Y))$.

A substitution applied to a term replaces variables by terms of smaller sorts. So for the substitution to be internal in the set of terms in lowest disambiguated form $LDT(Y)$, the rank of any operator has to remain the same, when the arguments of the operator are instantiated in a smaller lowest sort. For this requirement, we give a sufficient condition on the signature: the minimality defined below.

**Definition 2.** *An order-sorted signature $\Sigma$ is* **minimal** *if and only if for each $f : w \to s$ in $\Sigma$, if $f : w' \to s'$ is in $\Sigma$, then $w' = w$ or $w'$ is incomparable with $w$. In the same way, $\Sigma^\sharp$ is* **minimal** *if and only if for each $f_{w \to s}$ in $\Sigma^\sharp$, if $f_{w' \to s'}$ is in $\Sigma^\sharp$, then $w' = w$ or $w'$ is incomparable with $w$.*

Obviously, $\Sigma^\sharp$ is minimal if and only if $\Sigma$ is minimal. From now on, signatures $\Sigma$ are supposed to be minimal.

To become minimal, the signature of the example $EVEN$ has to be transformed. The operator "$is\_even$" has three possible ranks: $Int \to Bool$, $NzNeg \to Bool$ and $NzPos \to Bool$ such that $Int > NzNeg$ (and $Int > NzPos$ ), which contradicts the minimality condition. Minimality is obtained if the rank $Int \to Bool$ of "$is\_even$" is removed. But "$is\_even$" is no more defined for 0. In order to restore a full domain for "$is\_even$", one adds the rank $Zero \to Bool$. Thus, the term $is\_even(x)$ where the sort of $x$ is $Int$ is no more well-defined. We then have to split Rule (3) in

$$(\forall\, x : NzPos)\ is\_even(s(s(x))) \to is\_even(x)$$

and

$$is\_even(s(s(0))) \to is\_even(0).$$

The specification with a signature in minimal form is then (the differences with the previous specification $EVEN$ are bold-faced below):

$obj$ **MINIMAL_EVEN** $is$
$sorts\ Zero\ NzNeg\ Neg\ NzPos\ Pos\ Int\ Bool.$
$subsorts\ Zero < Neg < Int.$
$subsorts\ Zero < Pos < Int.$
$subsorts\ NzNeg < Neg.$
$subsorts\ NzPos < Pos.$

$op\ 0 :\to Zero.$
$op\ s : Pos \to NzPos.$
$op\ p : Neg \to NzNeg.$
$op\ true :\to Bool.$
$op\ false :\to Bool.$
**op is_even : Zero → Bool.**
$op\ is\_even : NzPos \to Bool.$
$op\ is\_even : NzNeg \to Bool.$
$op\ opposite : NzNeg \to NzPos.$

$rr\ is\_even(0) \to true.$ (1)
$rr\ is\_even(s(0)) \to false.$ (2)
$rr\ (\forall\, \mathbf{x} : \mathbf{NzPos})\ is\_even(s(s(x))) \to is\_even(x).$ (3)
$rr\ \mathbf{is\_even}(\mathbf{s}(\mathbf{s}(\mathbf{0}))) \to \mathbf{is\_even}(\mathbf{0}).$ (4)
$rr\ (\forall\, y : NzNeg)\ is\_even(y) \to is\_even(opposite(y)).$ (5)
$rr\ opposite(p(0)) \to s(0).$ (6)
$rr\ (\forall y\ : NzNeg)\ opposite(p(y)) \to s(opposite(y)).$ (7)
$jbo$

When the signature is finite (i.e. when the set of sorts is finite, the set of operators is finite and the set of ranks of any operator is finite), the problem of transforming a signature into a minimal one is decidable. We are now finishing designing algorithms for automatically performing this transformation, as well as the transformation of corresponding rewrite rules.

**Definition 3.** *If $\Sigma$ is a minimal signature, a* **substitution** *on $LDT(Y)$ is an S-sorted function $\sigma : X \to LDT(Y)$, that can be extended in a unique way into $\sigma : LDT(X) \to LDT(Y)$.*

**Definition 4. Disambiguated rewriting***: Let $R$ be a rewrite system defined on $LDT(Y)$. A term $t \in LDT(Y)$ rewrites to $t'$ with a rewrite rule $((\forall X)l \to r)$ in $R$ at occurrence $\omega$, which is denoted $t \hookrightarrow_Y^R t' = t[\omega \leftarrow \sigma r]$ whenever*

1. *there is a match $\sigma : X \to LDT(Y)$ from $l$ to $t$ at occurrence $\omega$ (a substitution $\sigma$ such that $\sigma l = t_{|\omega}$)*
2. *$t[\omega \leftarrow \sigma r]$ is a well-formed term of LDT(Y).*

Note that this definition is different from order-sorted rewriting on disambiguations developed in $OBJ3$ [13]. Sort-decreasingness can be defined on $LDT(Y)$ in the following way.

**Definition 5.** *A term rewriting system $R$ on $LDT(Y)$ is* **sort-decreasing** *if each rule of $R$ is, i.e., if for any rule $((\forall X)l \to r)$ in $R$, for any substitution $\sigma$ of $LDT(Y)$, we have $S(\sigma l) \geq S(\sigma r)$.*

Remark that when the system $R$ is sort-decreasing on $LDT(Y)$, then (1) $\Rightarrow$ (2) in the definition of disambiguated rewriting.

A lemma is now given, which avoids the need to ensure sort-decreasingness of disambiguated term rewriting systems, when they are deduced from sort-decreasing order-sorted ones.

**Lemma 2.** *Let $R$ be an order-sorted set of rules $(l_i \to r_i,\ i \in [1..n])$ on $\mathcal{T}_\Sigma(Y)$. If $R$ is sort-decreasing on $\mathcal{T}_\Sigma(Y)$, then $(LD(l_i) \to LD(r_i),\ i \in [1..n])$ is sort-decreasing on $LDT(Y)$.*

Proofs are omitted for lack of space. The reader can find them in [4].

The following theorem is a key of our termination proof method. It asserts that an order-sorted rewriting relation is included in the disambiguated relation, which is built from it. For proving termination of an order-sorted term rewriting system $R$ on $\mathcal{T}_\Sigma(Y)$, it then becomes enough to prove termination of its disambiguated form $LD(R)$ on $LDT(Y)$.

**Theorem 2.** *Let $R$ be an order-sorted term rewriting system and $t$ and $t'$ two terms of $\mathcal{T}_\Sigma(Y)$. Then $t \to^R t'$ implies $LD(t) \hookrightarrow^{LD(R)} LD(t')$.*

*Proof.* It lies on the fact that $\Sigma$ is minimal and $R$ is sort-decreasing. For a complete proof, see [4].

The signature of $MINIMAL\_EVEN$ is regular, and its rules are sort-decreasing. Its disambiguated presentation is (the differences with the previous specification $MINIMAL\_EVEN$ are bold-faced):

*obj* **UNAMBIGUOUS_MINIMAL_EVEN** *is*
*sorts* $Zero\ NzNeg\ Neg\ NzPos\ Pos\ Int\ Bool.$
*subsorts* $Zero < Neg < Int.$
*subsorts* $Zero < Pos < Int.$
*subsorts* $NzNeg < Neg.$
*subsorts* $NzPos < Pos.$

*op* $\mathbf{0}_{\to \mathbf{Zero}} :\to Zero.$
*op* $\mathbf{s}_{\mathbf{Pos} \to \mathbf{NzPos}} : Pos \to NzPos.$
*op* $\mathbf{p}_{\mathbf{Neg} \to \mathbf{NzNeg}} : Neg \to NzNeg.$
*op* $\mathbf{true}_{\to \mathbf{Bool}} :\to Bool.$
*op* $\mathbf{false}_{\to \mathbf{Bool}} :\to Bool.$
*op* $\mathbf{is\_even}_{\mathbf{Zero} \to \mathbf{Bool}} : Zero \to Bool.$

$op\ \mathbf{is\_even_{NzPos\to Bool}} : NzPos \to Bool.$
$op\ \mathbf{is\_even_{NzNeg\to Bool}} : NzNeg \to Bool.$
$op\ \mathbf{opposite_{NzNeg\to NzPos}} : NzNeg \to NzPos.$

$rr\ \mathbf{is\_even_{Zero\to Bool}}(\mathbf{0_{\to Zero}}) \to \mathbf{true_{\to Bool}}.$ (1)
$rr\ \mathbf{is\_even_{NzPos\to Bool}}(\mathbf{s_{Pos\to NzPos}}(\mathbf{0_{\to Zero}})) \to \mathbf{false_{\to Bool}}.$ (2)
$rr\ (\forall\ x : NzPos)\ \mathbf{is\_even_{NzPos\to Bool}}(\mathbf{s_{Pos\to NzPos}}(\mathbf{s_{Pos\to NzPos}}(\mathbf{x})))$
$\to \mathbf{is\_even_{NzPos\to Bool}}(\mathbf{x}).$ (3)
$rr\ \mathbf{is\_even_{NzPos\to Bool}}(\mathbf{s_{Pos\to NzPos}}(\mathbf{s_{Pos\to NzPos}}(\mathbf{0})))$
$\to \mathbf{is\_even_{Zero\to Bool}}(\mathbf{0}).$ (4)
$rr\ (\forall\ y : NzNeg)\ \mathbf{is\_even_{NzNeg\to Bool}}(\mathbf{y})$
$\to \mathbf{is\_even_{NzPos\to Bool}}(\mathbf{opposite_{NzNeg\to NzPos}}(\mathbf{y})).$ (5)
$rr\ \mathbf{opposite_{NzNeg\to NzPos}}(\mathbf{p_{Neg\to NzNeg}}(\mathbf{0_{\to Zero}})) \to \mathbf{s_{Pos\to NzPos}}(\mathbf{0_{\to Zero}}).$ (6)
$rr\ (\forall y\ : NzNeg)\ \mathbf{opposite_{NzNeg\to NzPos}}(\mathbf{p_{Neg\to NzNeg}}(\mathbf{y}))$
$\to \mathbf{s_{Pos\to NzPos}}(\mathbf{opposite_{NzNeg\to NzPos}}(\mathbf{y})).$ (7)
$jbo$

For more readability, ranks are omitted in unambiguous operators (non overloaded operators). The previous set of rules is then written:

$rr\ is\_even_{Zero\to Bool}(0) \to true.$ (1)
$rr\ is\_even_{NzPos\to Bool}(s(0)) \to false.$ (2)
$rr\ (\forall\ x : NzPos)\ is\_even_{NzPos\to Bool}(s(s(x))) \to is\_even_{NzPos\to Bool}(x).$ (3)
$rr\ is\_even_{NzPos\to Bool}(s(s(0))) \to is\_even_{Zero\to Bool}(0).$ (4)
$rr\ (\forall\ y : NzNeg)\ is\_even_{NzNeg\to Bool}(y)$
$\to is\_even_{NzPos\to Bool}(opposite(y)).$ (5)
$rr\ opposite(p(0)) \to s(0).$ (6)
$rr\ (\forall y\ : NzNeg)\ opposite(p(y)) \to s(opposite(y)).$ (7)

Hence, for proving the termination of the term rewriting system of the specification *MINIMAL_EVEN*, it is sufficient to prove the termination of the term rewriting system of the specification *UNAMBIGUOUS_MINIMAL_EVEN*.

## 3 Extended LPO is a simplification ordering

We are now ready to define our new ordering on $LDT(Y)$ called Order-sorted Ordering ($OSO$ in short). As suggested by the example in Section 1, it is an extension of $LPO$ [12].

**Definition 6. Lexicographic Path Ordering (LPO)**: *Let be $>_\Sigma$ a partial ordering on $\Sigma$ called precedence. The LPO on $\mathcal{T}_\Sigma(Y)$ is defined by: $t = f(t_1...t_m)\ >_{LPO}\ t' = g(t'_1...t'_n)$ iff:*

1. *$f = g$ and $t_1...t_m >^{lex}_{LPO} t'_1...t'_n$ and for each $j\ \in\ [1..n]$, $t >_{LPO} t'_j$ ($>^{lex}_{LPO}$ is the lexicographic extension of $>_{LPO}$) or else*
2. *$f >_\Sigma g$ and for each $j\ \in\ [1..n]$, $t >_{LPO} t'_j$ or else*
3. *there exists $i\ \in\ [1..m]$ such that $t_i >_{LPO} t'$ or $t_i = t'$*

**Definition 7.** *The Order-sorted Ordering OSO is the LPO on $LDT(Y)$.*

The precedence for $OSO$ is denoted by $>_{\Sigma^\sharp}$, provided the signature of $LDT(Y)$ is $\Sigma^\sharp$. We now aim to prove that $OSO$ is a simplification ordering on $LDT(Y)$, i.e., it has the replacement property, the subterm property and is stable by substitution. These three notions will be recalled in the following. Note that the deletion property [2] is not needed since the operators have fixed arity in $LDT(Y)$.

**Definition 8. Stability by substitution**: *An ordering $>$ on $LDT(Y)$ is stable by substitution iff for each $t$ and $t'$ in $LDT(Y)$, for each substitution $\sigma$ of $LDT(Y)$: $t > t' \Rightarrow \sigma t > \sigma t'$.*

**Proposition 1.** *Let us extend the precedence on $\Sigma^\sharp$ to $\Sigma^\sharp \cup Y$ in such a way that:*

**(Var 1)** *$x \bowtie y$ for $x, y \in Y$ (where $\bowtie$ means 'not comparable with')*
**(Var 2)** *$x \bowtie f_{s_1 \ldots s_m \to s}$ for $x \in Y$ and $f_{s_1 \ldots s_m \to s} \in \Sigma^\sharp$.*

*If in addition, $\Sigma^\sharp$ is minimal, then the OSO on $LDT(Y)$ is stable by substitution.*

*Proof.* The proof is by structural induction on $LDT(Y)$. The minimality of the signature is here a key property.

Let us now ensure the replacement property. We assume from now on that (Var 1) and (Var 2) are fulfilled for any precedence defining $OSO$.

**Proposition 2.** *The OSO has the replacement property on $LDT(Y)$, which means that for any $t$ and $t'$ of $LDT(Y)$, then $t >_{OSO} t'$ implies $h_{w\to s}(..t..) >_{OSO} h_{w\to s}(..t'..)$ for any context $h_{w\to s}(.. ..)$ such that $h_{w\to s}(..t..)$ and $h_{w\to s}(..t'..)$ are in $LDT(Y)$.*

*Proof.* By hypothesis, $t >_{OSO} t'$. Then, $..t.. >^{lex}_{OSO} ..t'..$ by definition of the lexicographic ordering. Let us use Case 1 of the definition of $OSO$ to conclude that $h_{w\to s}(..t..) >_{OSO} h_{w\to s}(..t'..)$.

**Proposition 3.** *The OSO is a strict ordering and has the subterm property.*

We thus can state the following theorem.

**Theorem 3.** *If $\Sigma^\sharp$ is minimal, the OSO is a simplification ordering on $LDT(Y)$.*

## 4 Termination theorem

Recall the first termination theorem of [2], which our work is based on.

**Theorem 4 (Dershowitz).** *A finite rewriting system $R$ over a set of terms $T$ is terminating if there exists a simplification ordering $>$ on $T$ such that for each rule $l \to r$ of $R$, we have $l > r$.*

Let us now establish that this theorem can be extended to $LDT(Y)$.

**Theorem 5.** *A finite rewriting system $R$ on the set $LDT(Y)$, whose signature $\Sigma^\sharp$ is minimal, is terminating if there exists a simplification ordering $>$ on $LDT(Y)$ such that for each rule $l \to r$ of $R$, we have $l > r$.*

The proof of this theorem is based on Kruskal's theorem, on the embedding lemma, and on the fact that if $g > d$ for each rule $g \to d$ of $R$, then $t \hookrightarrow_R t'$ implies $t > t'$. The proof of Kruskal's theorem is using general reasoning on term sequences and extracted subterm subsequences. It thus doesn't depend of the set of terms it is applied on. Hence, we claim that it is available on order-sorted term algebras. Both last properties, however, have to be carefully examined, because of the sort mechanism on terms.

**Definition 9.** *The homeomorphic embedding relation $\trianglerighteq$ on terms in $LDT(Y)$ is defined by: $t = f_{s_1,s_2\ldots s_m\to s}(t_1 \ldots t_m) \trianglerighteq t' = g_{s'_1,s'_2\ldots s'_m\to s'}(t'_1 \ldots t'_n)$ iff:*

1. *$f_{s_1,s_2\ldots s_m\to s} = g_{s'_1,s'_2\ldots s'_m\to s'}$ and for each $i \in [1..m]$, $t_i \trianglerighteq t'_i$ or else*

*2. there exists* $i \in [1..m]$ *such that* $t_i \trianglerighteq t'$

**Lemma 3. (Embedding lemma)** *Let* $t$ *and* $t'$ *be terms in* $LDT(Y)$ *and let* $\Sigma^\sharp$ *be minimal. If* $t \trianglerighteq t'$*, then* $t \geq t'$ *in any simplification ordering* $>$ *over* $LDT(Y)$ *(* $t \geq t'$ *means that* $t > t'$ *or* $t = t'$ *).*

*Proof.* The proof is by induction on the size (number of occurences of operators) of $t$.

**Lemma 4.** *Let* $>$ *be a simplification ordering on* $LDT(Y)$ *and let* $\Sigma^\sharp$ *be minimal. If* $l > r$ *for any rule* $l \to r$ *of a term rewriting system* $R$*, then* $t \hookrightarrow_R t'$ *implies* $t > t'$*.*

*Proof.* It requires sort-decreasingness of $R$ on $LDT(Y)$. Since we only consider sort-decreasing order-sorted rewriting systems on $\mathcal{T}_\Sigma(Y)$, their disambiguated form are sort-decreasing on $LDT(Y)$, by Lemma 2. Minimality is a key property for this proof as well as for the proof of Lemma 3.

**Proof of Theorem 5:** Let us suppose the existency of an infinite rewrite chain for $R$: $t_1 \hookrightarrow t_2 \hookrightarrow ... \hookrightarrow t_n \hookrightarrow ....$ By Kruskal's Theorem, there exists $i \leq j$ such that $t_i \trianglelefteq t_j$, and $t_i \leq t_j$ by Embedding Lemma. On the other hand, we get $t_1 > t_2 ... > t_n > ...$ by Lemma 4. Then $t_i > t_j$ by transitivity of $>$, which contradicts the asymmetry of $>$. □

## 5 Examples

We are now ready to prove termination of the rule system of the specification "$UNAMBIGUOUS\_MINIMAL\_EVEN$". With the precedence

$$is\_even_{Zero\to Bool} >_{\Sigma^\sharp} true,\ is\_even_{NzPos\to Bool} >_{\Sigma^\sharp} false,$$

$$is\_even_{NzPos\to Bool} >_{\Sigma^\sharp} is\_even_{Zero\to Bool},\ is\_even_{NzNeg\to Bool} >_{\Sigma^\sharp} is\_even_{NzPos\to Bool},$$
$$is\_even_{NzNeg\to Bool} >_{\Sigma^\sharp} opposite,\ opposite >_{\Sigma^\sharp} s,$$

one gets:

$$\begin{aligned}
&is_even_{Zero\to Bool}(0) >_{OSO} true,\\
&is_even_{NzPos\to Bool}(s(0)) >_{OSO} false,\\
&is_even_{NzPos\to Bool}(s(s(x))) >_{OSO} is_even_{NzPos\to Bool}(x),\\
&is_even_{NzPos\to Bool}(s(s(0))) >_{OSO} is_even_{Zero\to Bool}(0),\\
&is_even_{NzNeg\to Bool}(y) >_{OSO} is_even_{NzPos\to Bool}(opposite(y)),\\
&opposite(p(0)) >_{OSO} s(0),\\
&opposite(p(y)) >_{OSO} s(opposite(y)).
\end{aligned}$$

By Theorem 3 and Theorem 5, the rule system is terminating. Hence, the rule system of $MINIMAL\_EVEN$ is terminating, by Theorem 2.

Let us now look at a second example. We give a specification of the equality on naturals.

$$\begin{aligned}
&obj\ \textbf{NAT_EQUALITY}\ is\\
&sorts\ Zero\ NzNat\ Nat\ Bool.\\
&subsorts\ Zero < Nat.\\
&subsorts\ Nznat < Nat.\\
&\\
&op\ 0 : \to Zero.\\
&op\ true :\to Bool.\\
&op\ false :\to Bool.
\end{aligned}$$

$op\ s : Nat \to NzNat.$
$op\ equal : NzNat\ NzNat \to Bool.$
$op\ equal : Zero\ Zero \to Bool.$
$op\ equal : Nznat\ Zero \to Bool.$
$op\ equal : Zero\ NzNat \to Bool.$

$rr\ (\forall\ x : NzNat)\ equal(x, x) \to true$ (1)
$rr\ equal(0, 0) \to true.$ (2)
$rr\ (\forall\ x, x' : NzNat)\ equal(s(x), s(x')) \to equal(x, x').$ (3)
$rr\ (\forall\ x : NzNat)\ equal(s(x), s(0)) \to equal(x, 0).$ (4)
$rr\ (\forall\ x : NzNat)\ equal(s(0), s(x)) \to equal(0, x).$ (5)
$rr\ equal(s(0), s(0)) \to equal(0, 0).$ (6)
$rr\ (\forall\ y : NzNat)\ equal(y, 0) \to equal(0, y).$ (7)
$rr\ (\forall\ x : NzNat)\ equal(x, s(x)) \to false.$ (8)
$rr\ equal(0, s(0)) \to false.$ (9)
$jbo$

In the unsorted case, Rule (7) would not terminate: it corresponds to the commutativity of the operation "*equal*" with a null argument. But if $y$ is of sort $NzNat$, then the left-hand side cannot match the right-hand side, hence the rule is terminating. With an appropriate precedence (see [4]), *OSO* gives a termination proof for the above rewrite system.

Let us now illustrate that our method treats termination cases, that no known ordering could take into account till now.

Consider the rule $f(0, 1, x) \to f(x, x, x)$, extracted from a counter-example of Toyama to the termination of the direct sum of term rewriting systems [15]. It is clearly terminating but its termination cannot be proved with known unsorted orderings. We will see that *OSO* gives a termination proof for this rule, when typing it without lack of generality.

Let $s$ be the most general sort. Let $Zero$ be the sort of the constant 0, $One$ the sort of constant 1 and $N01$ the sort of any term except 0 and 1. To express the operator $f$ of the term $f(0, 1, x)$, one needs the three typed operators:

$$f :\ Zero\ One\ Zero \to s,\ f :\ Zero\ One\ One \to s,\ f :\ Zero\ One\ N01 \to s.$$

To express the operator $f$ of the term $f(x, x, x)$, one needs the three typed operators:

$$f :\ Zero\ Zero\ Zero \to s,\ f :\ One\ One\ One \to s,\ f :\ N01\ N01\ N01 \to s.$$

Hence the initial untyped rule is splitted in the three following forms:

$$f(0, 1, 0) \to f(0, 0, 0)$$
$$f(0, 1, 1) \to f(1, 1, 1)$$
$$(\forall\ x : N01)\ f(0, 1, x) \to f(x, x, x)$$

The complete specification equivalent to the initial rule is then:

$obj\ \mathbf{ZERO - ONE}\ is$
$sorts\ Zero\ One\ N01\ s.$
$subsorts\ Zero, One, N01 < s.$

$op\ 0 : \to Zero.$

$op\ 1 : \to One.$
$op\ f :\ Zero\ One\ Zero \to s.$
$op\ f :\ Zero\ One\ One \to s.$
$op\ f :\ Zero\ One\ N01 \to s.$
$op\ f :\ Zero\ Zero\ Zero \to s.$
$op\ f :\ One\ One\ One \to s.$
$op\ f :\ N01\ N01\ N01 \to s.$

$rr :\ f(0,1,0) \to f(0,0,0).$ (1)
$rr :\ f(0,1,1) \to f(1,1,1).$ (2)
$rr :\ (\forall\ x : N01)\ f(0,1,x) \to f(x,x,x).$ (3)
$jbo$

Note that its signature is regular and minimal. Its rules are sort-decreasing. Its disambiguated presentation is:

$obj$ **DISAMBIGUATED − ZERO − ONE** $is$
$sorts\ Zero\ One\ N01\ s.$
$subsorts\ Zero, One, N01 < s.$

$op\ 0 : \to Zero.$
$op\ 1 : \to One.$
$op\ f_{Zero\ One\ Zero \to s} :\ Zero\ One\ Zero \to s.$
$op\ f_{Zero\ One\ One \to s} :\ Zero\ One\ One \to s.$
$op\ f_{Zero\ One\ N01 \to s} :\ Zero\ One\ N01 \to s.$
$op\ f_{Zero\ Zero\ Zero \to s} :\ Zero\ Zero\ Zero \to s.$
$op\ f_{One\ One\ One \to s} :\ One\ One\ One \to s.$
$op\ f_{N01\ N01\ N01 \to s} :\ N01\ N01\ N01 \to s.$

$rr :\ f_{ZeroOneZero \to s}(0,1,0) \to f_{ZeroZeroZero \to s}(0,0,0).$ (1)
$rr :\ f_{ZeroOneOne \to s}(0,1,1) \to f_{OneOneOne \to s}(1,1,1).$ (2)
$rr :\ (\forall\ x : N01)\ f_{ZeroOneN01 \to s}(0,1,x) \to f_{N01N01N01 \to s}(x,x,x).$ (3)
$jbo$

Thus, with the precedence:

$$f_{ZeroOneZero \to s} >_{\Sigma^\sharp} f_{ZeroZeroZero \to s}$$
$$f_{ZeroOneOne \to s} >_{\Sigma^\sharp} f_{OneOneOne \to s}$$
$$f_{ZeroOneN01 \to s} >_{\Sigma^\sharp} f_{N01N01N01 \to s}$$

we have $l_i >_{OSO} r_i$ for the three rules $l_i \to r_i,\ i \in [1,2,3]$. Hence the term rewriting system of $DISAMBIGUATED - ZERO - ONE$ terminates. Thus, the term rewriting system of $ZERO - ONE$ also terminates.

## 6 Conclusion

In this work, we have given a tool for proving termination of order-sorted rewriting, focusing especially on cases where rules are terminating because of sorts arguments (that is, they would

not terminate in the unsorted case). An extension of Lexicographical Path Ordering was proposed, the Order Sorted Ordering, to prove termination of rewriting on disambiguated sets of terms. It has then been shown that for proving termination of an order-sorted term rewriting system, it is sufficient to prove termination of its corresponding disambiguated term rewriting system. A sufficient condition on the signature was assumed: the minimality, to define a disambiguated rewriting relation and to establish the validity of this extension. Thus, we have proved termination of examples, that could not be handled by any existing ordering. We will soon be able to propose algorithms for automatically transforming a signature into a minimal one, and for automatically adapting the corresponding set of rules.

## Acknowledgments

We would like to thank Pierre Lescanne for his support and fruitful discussions, and Hélène Kirchner for carefully reading a previous version of this paper.

## References


1. G. Bernot, M. Bidoit, and C. Choppy. Abstract data types with exception handling: an initial approach based on a distinction between exceptions and errors. *Theoretical Computer Science*, 46:13–45, 1986.
2. N. Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17:279–301, 1982.
3. N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3(1 & 2):69–116, 1987.
4. I. Gnaedig. Termination of rewriting in order-sorted algebras. Technical Report 91-R-108, Centre de Recherche en Informatique de Nancy, 1991.
5. I. Gnaedig. ELIOS-OBJ: Theorem proving in a specification language. In B. Krieg-Brückner, editor, *Proceedings of the 4th European Symposium on Programming*, volume 582 of *Lecture Notes in Computer Science*, pages 182–199. Springer-Verlag, February 1992.
6. I. Gnaedig, Claude Kirchner, and Hélène Kirchner. Equational completion in order-sorted algebras. *Theoretical Computer Science*, 72:169–202, 1990.
7. J. A. Goguen and J. Meseguer. Order-sorted algebra I: Partial and overloaded operations, errors and inheritance. Technical report, SRI International, Computer Science Lab, 1988. Given as lecture at a Seminar on Types, Carnegie-Mellon University, June 1983.
8. J. A. Goguen, J. Meseguer, and D. Plaisted. Programming with parameterized abstract objects in OBJ. *Theory And Practice of Software Technology*, pages 163–193, 1982.
9. John V. Guttag, James J. Horning, and J. M. Wing. Larch in five easy pieces. Technical report, Digital Systems Research Center, 1985.
10. J. Roger Hindley and Johnathan P. Seldin. *Introduction to Combinators and Lambda-calculus*. Cambridge University, 1986.
11. G. Huet and D. Oppen. Equations and rewrite rules: A survey. In R. V. Book, editor, *Formal Language Theory: Perspectives and Open Problems*, pages 349–405. Academic Press inc., 1980.
12. S. Kamin and J.-J. Lévy. Attempts for generalizing the recursive path ordering. *Inria, Rocquencourt*, 1982.
13. Claude Kirchner, Hélène Kirchner, and J. Meseguer. Operational semantics of OBJ-3. In *Proceedings of 15th International Colloquium on Automata, Languages and Programming*, volume 317 of *Lecture Notes in Computer Science*, pages 287–301. Springer-Verlag, 1988.
14. G. Smolka. Tel (version 0.9), report and user manual. SEKI report SR-87-11, Universität Kaiserslautern (Germany), 1988.
15. Y. Toyama. On the Church-Rosser property for the direct sum of term rewriting systems. *Journal of the ACM*, 34(1):128–143, January 1986.